# SWRL: A Semantic Web Rule Language

## Abstract Syntax, Human Readable Syntax, Interpreting Rules

The Semantic Web Rule Language (SWRL) is based on a combination of the OWL DL and OWL Lite sublanguages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language.

SWRL includes a high-level abstract syntax for Horn-like rules in both the OWL DL and OWL Lite sublanguages of OWL.

The proposed rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold.

Both the antecedent (body) and consequent (head) consist of zero or more atoms. An empty antecedent is treated as trivially true (i.e. satisfied by every interpretation), so the consequent must also be satisfied by every interpretation.

An empty consequent is treated as trivially false (i.e., not satisfied by any interpretation), so the antecedent must also not be satisfied by any interpretation. Multiple atoms are treated as a conjunction.

Atoms in these rules can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values.

# Abstract Syntax

The abstract syntax is specified here by means of a version of Extended BNF. Terminals are quoted; nonterminals are bold and not quoted. Alternatives are either separated by vertical bars (|) or are given in different productions.

Components that can occur at most once are enclosed in square brackets ([…]); components that can occur any number of times (including zero) are enclosed in braces ({…}). Whitespace is ignored in the productions here.

Names in the abstract syntax are RDF URI references. These names may be abbreviated into qualified names, using one of the namespace names shown on the next slide.

| Namespace name | Namespace |
| --- | --- |
| rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# |
| rdfs | http://www.w3.org/2000/01/rdf-schema# |
| xsd | http://www.w3.org/2001/XMLSchema# |
| owl | http://www.w3.org/2002/07/owl# |

## Rules

An OWL ontology in the abstract syntax contains a sequence of axioms and facts. Axioms may be of various kinds, e.g., subClass axioms and equivalentClass axioms. It is proposed to extend this with rule axioms.

***axiom*** ::= ***rule***

A rule axiom consists of an antecedent (body) and a consequent (head), each of which consists of a (posibly empty) set of atoms. A rule axiom can also be assigned a URI reference, which could serve to identify the rule.

***rule*** ::= '*Implies*(' [ ***URIreference*** ]
{ ***annotation*** } ***antecedent consequent*** ')'
***antecedent*** ::= '*Antecedent*(' { ***atom*** } ')'
***consequent*** ::= '*Consequent*(' { ***atom*** } ')'

Informally, a rule may be read as meaning that if the antecedent holds (is "true"), then the consequent must also hold. An empty antecedent is treated as trivially holding (true), and an empty consequent is treated as trivially not holding (false).

Rules with an empty antecedent can thus be used to provide unconditional facts; however such unconditional facts are better stated in OWL itself, i.e., without the use of the rule construct. Non-empty antecedents and consequents hold iff all of their constituent atoms hold, i.e., they are treated as conjunctions of their atoms.

***atom*** ::= ***description*** '(' ***i-object*** ')'

| ***dataRange*** '(' ***d-object*** ')'

| ***individualvaluedPropertyID*** '(' ***i-object i-object*** ')'

| ***datavaluedPropertyID*** '(' ***i-object d-object*** ')'

| *sameAs* '(' ***i-object i-object*** ')'

| *differentFrom* '(' ***i-object i-object*** ')'

| *builtIn* '(' ***builtinID*** { ***d-object*** } ')'

***builtinID*** ::= ***URIreference***

Atoms can be of the form C(x), P(x,y), sameAs(x,y) differentFrom(x,y), or builtIn(r,x,...) where C is an OWL description or data range, P is an OWL property, r is a built-in relation, x and y are either variables, OWL individuals or OWL data values, as appropriate.

In the context of OWL Lite, descriptions in atoms of the form C(x) may be restricted to class names.

Informally, an atom C(x) holds if x is an instance of the class description or data range C.

An atom P(x,y) holds if x is related to y by property P, an atom sameAs(x,y) holds if x is interpreted as the same object as y, an atom differentFrom(x,y) holds if x and y are interpreted as different objects, and builtIn(r,x,...) holds if the built-in relation r holds on the interpretations of the arguments.

Note that the sameAs and differentFrom two forms can be seen as "syntactic sugar": they are convenient, but do not increase the expressive power of the language (i.e., such (in)equalities can already be expressed using the combined power of OWL and rules without explicit (in)equality atoms).

Atoms may refer to individuals, data literals, individual variables or data variables. Variables are treated as universally quantified, with their scope limited to a given rule. As usual, only variables that occur in the antecedent of a rule may occur in the consequent (a condition usually referred to as "safety").

This safety condition does not, in fact, restrict the expressive power of the language (because existentials can already be captured using OWL someValuesFrom restrictions).

## Human Readable Syntax

While the abstract EBNF syntax is consistent with the OWL specification, and is useful for defining XML and RDF serialisations, it is rather verbose and not particularly easy to read. In the following we will, therefore, often use a relatively informal "human readable" form similar to that used in many published works on rules.

In this syntax, a rule has the form:

*antecedent* ⇒ *consequent*

where both antecedent and consequent are conjunctions of atoms written

a1 ∧ ... ∧ an.

Variables are indicated using the standard convention of prefixing them with a question mark (e.g., ?x). Using this syntax, a rule asserting that the composition of parent and brother properties implies the uncle property would be written:

$$parent(?x,?y) \wedge brother(?y,?z) \Rightarrow uncle(?x,?z)$$

In this syntax, built-in relations that are functional can be written in functional notation, i.e.,

op:numeric-add(?x,3,?z)

can be written instead as

?x = op:numeric-add(3,?z)

# Direct Model-Theoretic Semantics

The model-theoretic semantics for SWRL is a straightforward extension of the semantics for OWL. The basic idea is that we define **bindings**, extensions of OWL interpretations that also map variables to elements of the domain.

A rule is satisfied by an interpretation iff every binding that satisfies the antecedent also satisfies the consequent. The semantic conditions relating to axioms and ontologies are unchanged, e.g., an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology.

# Interpreting Rules

Given a datatype map D, an abstract OWL interpretation is a tuple of the form

I = <R, EC, ER, L, S, LV>

where R is a set of resources, LV ⊆ R is a set of literal values, EC is a mapping from classes and datatypes to subsets of R and LV respectively, ER is a mapping from properties to binary relations on R, L is a mapping from typed literals to elements of LV, and S is a mapping from individual names to elements of EC(owl:Thing).

To handle the built-in relations, we augment the datatype map to map the built-in relations to tuples over the appropriate sets. That is, *op:numeric-add* is mapped into the triples of numeric values that correctly interpret numeric addition.

Given an abstract OWL interpretation I, a binding B(I) is an abstract OWL interpretation that extends I such that S maps i-variables to elements of EC(owl:Thing) and L maps d-variables to elements of LV respectively.

An atom is satisfied by an interpretation I under the conditions given in the ***Interpretation Conditions Table***, where C is an OWL DL description, D is an OWL DL data range, P is an OWL DL individualvalued property, Q is an OWL DL datavalued property, f is a built-in relation, x,y are variables or OWL individuals, and z is a variable or an OWL data value.

## *Interpretation Conditions Table*

| **Atom** | **Condition on Interpretation** |
|---|---|
| *C(*x*)* | S(x) ∈ EC(C) |
| *D(*z*)* | S(z) ∈ EC(D) |
| *P(*x,y*)* | <S(x),S(y)> ∈ ER(P) |
| *Q(*x,z*)* | <S(x),L(z)> ∈ ER(Q) |
| *sameAs(*x,y*)* | S(x) = S(y) |
| *differentFrom(*x,y*)* | S(x) ≠ S(y) |
| *builtIn(*r,z1,...,zn*)* | <S(z1),...,S(zn)> ∈ D(f) |

A binding B(I) satisfies an antecedent A iff A is empty or B(I) satisfies every atom in A. A binding B(I) satisfies a consequent C iff C is not empty and B(I) satisfies every atom in C. A rule is satisfied by an interpretation I iff for every binding B such that B(I) satisfies the antecedent, B(I) also satisfies the consequent.

Note that rule annotations have no semantic consequences and neither do the URI references associated with rules. This is different from the situation for OWL itself, where annotations do not have semantic consequences.

The semantic conditions relating to axioms and ontologies are unchanged. In particular, an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology; an ontology is consistent iff it is satisfied by at least one interpretation; an ontology $O_2$ is entailed by an ontology $O_1$ iff every interpretation that satisfies $O_1$ also satisfies $O_2$.

# Example Rules

**Example 1**

A simple use of these rules would be to assert that the combination of the hasParent and hasBrother properties implies the hasUncle property. Informally, this rule could be written as:

$$\text{hasParent}(?x1,?x2) \wedge \text{hasBrother}(?x2,?x3) \Rightarrow \text{hasUncle}(?x1,?x3)$$

In the abstract syntax the rule would be written like:

```
Implies(Antecedent(hasParent(I-variable(x1)
                             I-variable(x2))
                   hasBrother(I-variable(x2)
                              I-variable(x3)))
        Consequent(hasUncle(I-variable(x1)
                            I-variable(x3))))
```

From this rule, if John has Mary as a parent and Mary has Bill as a brother then John has Bill as an uncle.

**Example 2**

An even simpler rule would be to assert that Students are Persons, as in

Student(?x1) ⇒ Person(?x1)

Implies(Antecedent(Student(I-variable(x1)))
Consequent(Person(I-variable(x1))))

However, this kind of use for rules in OWL just duplicates the OWL subclass facility.

It is logically equivalent to write instead

Class(Student partial Person)

or

SubClassOf(Student Person)

which would make the information directly available to an OWL reasoner.

# Usage Suggestions

## Extensibility and Interoperability Cautions

If users are making extensive use of rules, they may want to restrict the form or expressiveness of the rules they employ, in order to increase interoperability, reusability, extensibility, computational scaleability, or ease of implementation.

A useful restriction in the form of the rules is to limit antecedent and consequent classAtoms to be named classes, where the classes are defined purely in OWL (in the same document or in external OWL documents).

Adhering to this format makes it easier to translate rules to or from existing (or future) rule systems, including:

1. Prolog;
2. production rules (descended from OPS5);
3. event-condition-action rules; and
4. SQL (where views, queries, and facts are all rules).

Adhering to this form also maximises reuse and interoperability of the ontology knowledge in the rules with other OWL-speaking systems that do not necessarily support SWRL.

Users also may want to restrict the expressiveness of the OWL classes and descriptions appearing in rules. One useful restriction on expressivity is *Description Logic Programs* which, e.g., prohibits existentially-quantified knowledge in consequents.

Suitably-restricted SWRL rules can be straightforwardly extended to enable procedural attachments and/or nonmonotonic reasoning (negation-as-failure and/or prioritised conflict handling) of the kinds supported in CCI rule systems. Such adherence may thus facilitate combining SWRL knowledge with knowledge from those other rules languages.